

EV316935335

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**APPLICATION FOR LETTERS PATENT**

**SYSTEMS AND METHODS FOR MANAGING  
FRAME RATES DURING MULTIMEDIA  
PLAYBACK**

Inventor(s):

Charles R. Kellner, Jr.

William R. Sanders

Darren R. Davis

ATTORNEY'S DOCKET NO. MS1-1683US

## **TECHNICAL FIELD**

The systems and methods described herein relate to multimedia playback and, more particularly, to managing frame rates of the multimedia playback.

## **BACKGROUND**

Today's computers are versatile machines capable of performing many tasks that traditionally require many separate machines. For example, general purpose personal computers are increasingly being used for playing back multimedia data, which usually require dedicated video viewing electronics, such as VCD players, DVD players, and the like. Multimedia data are often compressed for ease of transmission and storage. Substantial processing power and memory capabilities are required to decompress and render multimedia data. These requirements for multimedia playback can be taxing even for powerful personal computers.

Multimedia playback is a familiar and desirable feature of today's small handheld computers and other portable consumer electronic devices. The design of these computers places priorities on portability and economy. These priorities often result in reductions in processing power and memory capacity. Also, many of today's portable consumer electronic devices, including handheld computers, typically lack hardware audio/video acceleration. As a result, handheld computers and other portable consumer electronic devices typically offer a small fraction of the multimedia playback capabilities possessed by a typical personal computer. Still, users are increasingly demanding and expecting their portable consumer

1 electronic devices to playback the same multimedia data that can be handled by  
2 their more powerful personal computers.

3 The current trends of multimedia data are to use higher frame rates and  
4 resolution to increase the quality of the playback while using more aggressive  
5 video compression methods to maintain reasonable requirements for transmission  
6 and storage of the data. Achieving smooth playback of computationally intensive  
7 multimedia data is challenging on a computer having insufficient processing  
8 power. Conventional audio/video synchronization methods do not deal adequately  
9 with this problem. In fact, these conventional methods tend to degrade the quality  
10 of the multimedia playback by dropping frames at the most inopportune moments,  
11 such as when the viewer's full attention is attracted by quick action or scene  
12 changes.

13 Thus, there is a need for a system that enables multimedia playback on  
14 computers with limited processing power without substantially degrading the  
15 quality of the playback.

## 16 17 **SUMMARY**

18 The systems and methods described herein are directed at managing frame  
19 rates during multimedia playback. In one aspect, the systems and methods  
20 determine the ideal playback timing associated with video data. If the actual  
21 playback timing of the video data lags the ideal playback timing, the frame rates  
22 associated with the video data are smoothly varied to recover toward the ideal  
23 playback timing.

1 In another aspect, an average delay associated with the playback of multiple  
2 frames of the video data is computed. The frame rates are varied in accordance  
3 with a smoothing function having the average delay as a variable.

4 In yet another aspect, the systems and methods apply an iterative frame-  
5 dropping algorithm to determine whether a particular frame in the video data  
6 should be dropped. The iterative frame-dropping algorithm takes previous frames  
7 into account to smoothly vary the frame rate.

### 8 9 **BRIEF DESCRIPTION OF THE DRAWINGS**

10 Fig. 1 is a graphical representation of an example multimedia player;

11 Fig. 2a is a graphical representation of a portion of compressed video data;

12 Fig. 2b is a graphical representation of a processing timeline associated  
13 with a P-frame in compressed video data;

14 Fig. 3 shows some exemplary graphs associated with the processing and  
15 rendering of compressed video data;

16 Fig. 4 is an operational flow diagram of an example process for processing  
17 multimedia data;

18 Fig. 5 is an operational flow diagram of an exemplary process for  
19 synchronizing video data; and

20 Fig. 6 illustrates an exemplary computer within which the systems and  
21 methods for managing frame rates during multimedia playback can be either fully  
22 or partially implemented.

## **DETAILED DESCRIPTION**

Multimedia data typically include compressed audio and video data. Multimedia playback requires both audio and video data to be decompressed. Generally, audio data compression techniques require less processing power for decompression than video data. For video data, the processing power needed for decompression typically varies with the complexity of the multimedia data. In particular, video data having images that change significantly from frame to frame, such as those involving action sequences or a scene change, require more processing power to decompress than video data with stable images. Multimedia playback on a computer with limited processing capabilities often results in delays in processing the video data relative to the ideal playback timing, such as the playback timing for the audio data. One popular method for compensating for such delays is to drop (e.g., delete) frames in the video data.

Human perception of motion is an important factor to consider in determining how frames should be dropped. In this regard, some studies have suggested that lower frame rates are interpreted by an interpolation mechanism in the brain that reconstructs smooth motion using visible frames as reference points. For example, see Badler and Tsotsos, "Motion: Representation and Perception," *Proceedings of the ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, 1983.

Predictable intervals between visual frames may lead to more successful reconstruction of smooth motion by the human brain. This mechanism may be more successful when the intervals between frames are more predictable. In particular, less visual processing is needed if the frame rates vary in a smooth

1 manner. When this happens, the viewer experiences a more relaxed feeling, which  
2 may produce the perception of smoother motion.

3 Thus, the systems and methods discussed herein provide for managing  
4 frame rates during multimedia playback. These systems and methods manage  
5 frame rates differently from the ways in which convention systems manage frame  
6 rates. For example, many conventional systems abruptly drop frames in a video  
7 stream when there is a significant delay in processing video data. These systems  
8 tend to cause abruptly changing frame rates and choppy images, resulting in a poor  
9 viewing experience.

10 In contrast, the described systems and methods implement a frame-  
11 dropping mechanism that controls the frame rates in accordance with a smoothing  
12 function. This smoothing function may include the video processing delay as a  
13 variable. Smoothly varying the frame rates results in more predictable intervals  
14 between frames and, thus, enhances the viewer's experience. Controlling the  
15 frame rates in this manner also improves the viewer's experience when the frame  
16 rates are low. Specifically, the perception of audio/video synchronization is  
17 necessarily more approximate as the interval between frames becomes longer. The  
18 synchronization delay is less apparent when it is a smoothing function of the  
19 actual displayed frame interval.

20 Fig. 1 is a graphical representation of an example multimedia player 100.  
21 Multimedia player 100 is a computer-executable component configured to process  
22 and display multimedia data 105. Generally, multimedia data 105 is compressed  
23 to reduce storage and facilitate transmission. Multimedia data 105 may include  
24 many types of data, such as compressed audio data 106, compressed video data  
25

1 107, and the like. For purposes of discussion, multimedia data 105 is illustrated to  
2 include both compressed audio data 106 and compressed video data 107.

3 Multimedia player 100 may access multimedia data 105 by accessing a  
4 computer-readable media where the data are stored. Such computer-readable  
5 media may include hard drives, DVDs, memory, and the like. Multimedia data  
6 105 may also be streamed over a network, such as the Internet, and be accessible  
7 by multimedia player 100 in real time. Multimedia player 100 is configured to  
8 parse multimedia data 105 into compressed audio data 106 and compressed video  
9 data 107 and to process them to obtain processed audio data 156 and processed  
10 video data 157. Multimedia player 100 is also configured to interact with other  
11 components in a computer to playback processed audio data 156 and processed  
12 video data 157 to a viewer.

13 Multimedia player 100 may include many logical components. For ease of  
14 discussion, multimedia player 100 is illustrated in Fig. 1 to include presentation  
15 clock 110, audio decoder 112, audio scheduler 114, audio renderer 116, video  
16 decoder 122, video renderer 126, and Synchronizer 135. In practice, multimedia  
17 player 100 may include more or less logical components than those shown in Fig.  
18 1.

19 Audio decoder 112 is configured to decode compressed audio data 106 and  
20 associated information as part of the audio data stream. Audio decoder 112 is  
21 typically configured to handle various types of audio data format. Audio decoder  
22 112 may use one or more types of audio compressor/decompressor (CODEC) to  
23 process compressed audio data 106. Typically the audio data stream also includes  
24 information about the ideal playback timing of the audio data.  
25

1       Presentation clock 110 is a reference clock for managing the playback  
2 timing of multimedia data presentation. Presentation clock 110 may obtain  
3 playback timing information from a system timer, and/or other timing sources.  
4 Audio scheduler 114 is configured to manage the playback timing of audio data.  
5 Audio scheduler 114 may interact with presentation clock 110 to obtain the ideal  
6 multimedia presentation timing and to schedule the playback of the audio data in  
7 accordance with the ideal audio playback timing. Audio scheduler 114 may also  
8 interact with presentation clock 110 to provide synchronization feedback to  
9 guarantee that the presentation clock and audio hardware maintain consistency.  
10 Audio renderer 116 is configured to create processed audio data 156, which is in a  
11 format that is usable by other audio related components in a computer. Audio  
12 renderer 116 may be configured to interact with the other software and hardware  
13 components to facilitate the playback of processed audio data 156 to users.

14       Video decoder 122 is configured to decode compressed video data 107 and  
15 associated information that may be part of the video stream. Video decoder 122  
16 may use one or more video CODEC's to process compressed video data 107.  
17 Processing of video data will be discussed in more detail below. Briefly stated,  
18 video decoder 122 is configured to construct sequential frames of video from  
19 compressed video data 107. In particular, video decoder 122 is configured to  
20 handle compressed video data that require the construction of a frame using  
21 information from a previous frame. In one embodiment, the video decoder 122 is  
22 configured to process a current frame to obtain just enough information for  
23 constructing the next frame, without actually constructing the current frame.  
24 Processing the current frame in this manner enables the current frame to be  
25



1 dropped without detrimentally affecting the processing of subsequent frames. In  
2 addition to video frames, video decoder typically decodes information about the  
3 ideal presentation time for video frames from the video data stream.

4 Video renderer 126 is configured to create processed video data 157, which  
5 is in a format that is usable by other video related components in a computer.  
6 Video renderer 126 is typically configured to interact with the other software and  
7 hardware components to facilitate the playback of processed video data 157.

8 Synchronizer 135 is a computer-executable component configured to  
9 control the frame rates for multimedia playback. In particular, Synchronizer 135 is  
10 configured to determine the preferred playback timing for frames of video. In one  
11 embodiment, Synchronizer 135 is configured to interact with presentation clock  
12 110 to obtain the multimedia presentation playback timing and uses that timing to  
13 schedule the playback for the frames. Synchronization of audio and video  
14 presentation is guaranteed by each data stream using a common presentation clock  
15 as the reference timer. In many cases the presentation clock is synchronized to the  
16 audio playback hardware. However, it is to be understood that presentation clock  
17 110 may determine the playback timing in other ways, especially for multimedia  
18 data that do not have an audio component. For example, presentation clock 110  
19 may obtain the ideal playback timing by determining such information directly  
20 from the system timer.

21 Synchronizer 135 may apply additional control to the frame rates in order  
22 to optimize video presentation in relation to presentation clock 110 to yield the  
23 preferred frame rate. Since Synchronizer 135 may be tightly coupled to the  
24 presentation clock 110 to determine video playback timing in an unconstrained  
25

environment, variability in presentation clock 110 may yield similar variability in the resulting preferred frame rate as determined by Synchronizer 135. Presentation clock 110 may include a filter configured to remove variability or noise to improve the performance of Synchronizer 135.

Synchronizer 135 is configured to interact with video decoder 122 and video renderer 126. Specifically, Synchronizer 135 is configured to obtain from video decoder 122 data associated with the processing of compressed video data 107 and from video renderer 126 data associated with the rendering of processed video data 157. Synchronizer 135 may also be configured to control how each frame is processed by video decoder 122, which enables Synchronizer 135 to drop frames when appropriate.

From the data obtained from video decoder 122 and video renderer 126, Synchronizer 135 is configured to determine whether the actual playback timing associated with multimedia data 105 lags the ideal playback timing. Specifically, Synchronizer 135 calculates a delay time associated with the playback of multimedia data 105. Synchronizer 135 then applies a function to determine whether a particular frame should be dropped. This function is a smoothing function of the delay. Thus, even if frames are dropped during the playback of multimedia data 105, the frame rates of the video vary smoothly with the delay.

In one embodiment, the delay (Delay) is calculated by formula 1 below:

$$(1) \text{ Delay} = \text{DecodeTime} - \text{PresentationTime}$$

1 where “PresentationTime” is the ideal playback time of a current video frame and  
2 “DecodeTime” is the actual completion time of the decoding process associated  
3 with the current frame.

4 A running average of Delay (AvgDelay) may be calculated by formula 2  
5 below:

$$7 \quad (2) \text{ AvgDelay} = (\text{Delay} + (\text{AvgDelay} * (\text{Aperture} - 1))) / \text{Aperture}$$

8  
9 where the Aperture is a parameter associated with the smoothness of the frame  
10 rate variation and represents the number of frames that are used for averaging. In  
11 one embodiment, an Aperture of 6 frames generally yields good results. This  
12 running average is used to control a frame-dropping algorithm that includes a  
13 rasterization algorithm. Briefly stated, rasterization involves one whole number  
14 being added to another and compared to a third. Overflow causes the third  
15 number to be subtracted and triggers a secondary behavior. Bresenham iterative  
16 line drawing algorithm is an example of a rasterization algorithm. One  
17 embodiment of such a frame-dropping algorithm is illustrated by the following:

```
18      FrameSkipFactor = (AvgDelay / 4) + IdealFrameRate -  
19 TolFactor;  
20      if (FrameSkipFactor > IdealFrameRate)  
21      {  
22          Iterator += IdealFrameRate;  
23          if (Iterator >= FrameSkipFactor)  
24              Iterator -= FrameSkipFactor;  
25          else  
26              (drop this frame)  
27      }
```

1 where the FrameSkipFactor is a parameter for determining whether a frame is to  
2 be dropped. The IdealFrameRate is a parameter associated with the ideal frame  
3 rate (e.g. number of milliseconds per frame) for the multimedia data. The  
4 TolFactor is a tolerance factor to allow for expected variability in the system timer.  
5 In one embodiment, a TolFactor of 20 milliseconds is representative of an  
6 exemplary system in accordance with this embodiment. The Iterator is a  
7 parameter for comparing with the FrameSkipFactor to determine whether to drop a  
8 particular frame. The Iterator value is continuously maintained for analyzing each  
9 frame in the compressed video data 107.

10 As shown in the frame-dropping algorithm above, the FrameSkipFactor is  
11 calculated for a frame as a fraction of the Delay plus the IdealFrameRate minus  
12 the TolFactor. If the FrameSkipFactor is greater than the IdealFrameRate, it  
13 means that the video decoder is unable to keep up with the ideal playback timing  
14 of the multimedia data and the frame may have to be dropped. In this case, the  
15 IdealFrameRate is added to the Iterator, and the result is compared to the  
16 FrameSkipFactor. Overflow is normal and expected, and means that the frame  
17 should be shown. When the Iterator is less than the FrameSkipFactor, the frame is  
18 dropped. This algorithm evenly distributes dropped frames.

19 Multimedia player 100 may be configured to improve the performance of  
20 Synchronizer 135 by buffering video data. In one embodiment, rendered video  
21 frames may be stored in a buffer so that Synchronizer 135 is operating ahead of  
22 real time. Buffering enables Synchronizer 135 to maintain consistent performance  
23 even if Synchronizer 135 is lagging according to its internal clock. The lag would  
24 push Synchronizer 135 closer to real time instead of falling behind it.  
25

1 Fig. 2a is a graphical representation of a portion of compressed video data  
2 107. Fig. 2b is a graphical representation of a processing timeline associated with  
3 a P-frame 221 in compressed video data 107. As shown in Fig. 2a, compressed  
4 video data 107 may include intra-coded frame (I-frame) 201, predicted frames (P-  
5 frames) 231-232 and bidirectional frames (B-frames) 221-224.

6 I-frame 201 is a frame that is compressed without reference to any other  
7 frame in compressed video data 107. This allows the video data to be played from  
8 random positions and for fast forward/reverse. Generally, a multimedia player  
9 must start decoding from an I-frame in a sequence of frames. The frequency of I-  
10 frames is generally set when the multimedia stream is compressed or is determined  
11 by the encoding application in order to balance trade offs between number of bits  
12 required and complexity of the decoding operation. I-frames are generally  
13 encoded at a frequency of 1 every 4 to 8 seconds.

14 Each of the P-frames 231-232 is a frame that is coded as differences from  
15 the last I-frame or P-frame. The new P-frame is decoded by taking the last I-frame  
16 or P-frame and calculating the values of each new pixel in the new P-frame. A P-  
17 frame typically yields a compression ratio better than an I-frame and requires more  
18 processing power to decompress than the I-frame. Since the amount of  
19 compression depends on the amount of motion present in the P-frame, the  
20 processing power required for decompressing a P-frame depends on the  
21 complexity of the frame.

22 Each of the B-frames 221-224 is a frame that is coded as differences from  
23 the last or next I-frame or P-frame. B-frames require both previous and  
24 subsequent frames for correct decoding and typically give improved compression  
25

1 compared with P-frames. However, B-frames generally require more processing  
2 power to decode.

3 In Fig. 2b, a video decoder is currently processing P-frame 231 in  
4 compressed video data 107 at decoding time 245. However, multimedia data 107  
5 is actually required to be rendered at presentation time 246. Thus, a delay 249 is  
6 present and frames in compressed video data 107 may have to be dropped in  
7 accordance with the frame-dropping algorithm discussed above in conjunction  
8 with Fig. 1.

9 In one embodiment, Synchronizer 135 is configured to drop B-frames  
10 whenever a delay is present. Synchronizer 135 may also be configured to drop B-  
11 frames if it determines that there would be significant time savings over dropping  
12 the next I-frame or P-frame. In this event, the next frame is shown, even though it  
13 was scheduled to be dropped. Thus, B-frames 221-222 may be dropped to  
14 compensate for delay 249.

15 Next, P-frames may be dropped in accordance with the frame-dropping  
16 algorithm. For illustrative purposes, suppose that Synchronizer 135 determines  
17 that P-frame 231 is to be dropped. In one embodiment, to maintain the integrity of  
18 compressed video data 107, Synchronizer 135 may be configured to drop all of the  
19 P-frames and B-frames following P-frame 231 up to the next I-frame. In another  
20 embodiment, Synchronizer 135 may be configured to process P-frame 231 to  
21 obtain enough information to process P-frame 232.

22 As discussed above, Synchronizer 135 may use a frame-dropping algorithm  
23 that is averaged over time. Under some conditions, it is possible to catch up faster  
24 than the presentation time 246 intended for the compressed video data. In this  
25

1 case, the decision to drop the frame in accordance with the frame-dropping  
2 algorithm may be overridden. In another embodiment, exceptions are made to  
3 force the display of the next frame whenever the algorithm determines to skip  
4 beyond the next I-frame. In yet another embodiment, exceptions are made to force  
5 the next frame whenever there actually is no new frame after the current frame.  
6 This ensures that the last frame of video data will be shown, even if the frame-  
7 dropping algorithm determined otherwise.

8 Fig. 3 shows some exemplary graphs associated with the processing and  
9 rendering of compressed video data. Three exemplary frames 311-313 of a  
10 portion of compressed video data 107 are shown for illustrative purposes. Frame  
11 311 and frame 313 are the first frames of a sequence of frames that contain stable  
12 images, such as a landscape scene. Frame 312 is the first frame of a complex  
13 sequence of frames containing many rapidly changing images, such as an action  
14 scene.

15 Graph 320 illustrates the processing requirements associated with the  
16 processing and rendering of compressed video data 107. The processing  
17 requirements may include a variety of relevant parameters such as processor  
18 usage, memory usage, and the like. As shown in the figure, the processing  
19 requirements for processing frame 311 are substantial due to a scene change.  
20 Because the frame sequence associated with frame 311 contains stable images, the  
21 processing requirements for processing the subsequent frames in the frame  
22 sequence are comparatively lower. The processing requirements increase again  
23 when frame 312 are processed. However, because the frame sequence associated  
24 with frame 312 contains images that change substantially from frame-to-frame, the  
25

1 processing requirements for processing the subsequent frames remain at a higher  
2 level throughout the frame sequence. After processing frame 313, the processing  
3 requirements become comparatively low again because of the stable images in the  
4 frame sequence associated with frame 313.

5 Graph 330 shows the typical frame rates that can be attained using a  
6 conventional video processing algorithm. Generally, frame rates are limited by the  
7 processing power available in the system. Specifically, if the processing  
8 requirements for processing compressed video data 107 exceed the available  
9 processing power in the system, a delay would result. To compensate, frames  
10 associated with the video data would have to be dropped to realize the preferred  
11 playback timing of the video data, resulting in frame rates that are lower than the  
12 ideal frame rates associated with the video data. For purposes of discussion, the  
13 system associated with Fig. 3 does not have sufficient processing power to realize  
14 the ideal frame rates.

15 As shown in graphs 320 and 330, processing frame 311 results in an  
16 increase in processing requirements (graph 320) and a corresponding decrease in  
17 frame rates (graph 330). Because the conventional video processing algorithm  
18 associated with graph 330 is configured to abruptly drop frames when there is a  
19 significant delay, the frame rates attained by that algorithm also abruptly vary as a  
20 step function. The abruptly varying frame rates result in choppy images and  
21 severely degrade the viewer's experience. To make matter worse, the  
22 conventional video processing algorithm is not configured to recover from a  
23 severe delay, such as the delay resulting from processing the video sequence  
24 associated with frame 312. As shown in Fig. 3, the conventional video processing  
25



1 algorithm (graph 330) simply allows the frame rates to drop to zero, which is  
2 detrimental to the viewer's enjoyment.

3 In contrast, the frame dropping algorithm employed by Synchronizer 135 of  
4 the present invention varies the frame rates in accordance with a smoothing  
5 function. As shown in graph 340, Synchronizer 135 allows the frame rates to  
6 smoothly drop as a result of processing frame 311. The frame dropping algorithm  
7 of Synchronizer 135 enables the frame rates to smoothly recover for the remaining  
8 frames of the sequence. Significantly, the frame dropping algorithm enables  
9 Synchronizer 135 to maintain the frame rates above zero. Thus, frames in the  
10 sequence associated with frame 312 are still shown to the viewer with smoothly  
11 decreasing frame rates, which is an improvement in viewer's experience compared  
12 to a zero frame rate.

13 After processing the frame sequence associated with frame 312,  
14 Synchronizer 135 enables the frame rates to smoothly increase to higher values.  
15 Since the frame rates are increasing, it may be advantageous for the frames rate to  
16 rapidly jump to the highest value available. In one embodiment, Synchronizer 135  
17 may be configured to achieve this by enabling an abrupt frame rate increase as  
18 indicated by the aggressive frame rate curve 345 shown as a dash line.

19 Fig. 4 is an operational flow diagram of an exemplary process 400 for  
20 processing multimedia data. Process 400 may be employed by a multimedia  
21 player to handle multimedia data. Moving from a start block, the process goes to  
22 block 410 where multimedia data is received. The multimedia data may include a  
23 video component but may also include other components, such as an audio  
24 component.  
25

1       At block 413, the ideal playback timing of the video data is determined.  
2       The information about the ideal playback timing is typically encoded in the  
3       multimedia data.

4       At block 415, the video data are processed, which may include operations  
5       associated with decompression and rendering. The video data are typically  
6       processed one frame at a time. At block 420, a delay is calculated. Typically, if a  
7       system does not have sufficient processing resources, the video data cannot be  
8       processed fast enough to keep pace with the ideal playback timing of the video  
9       data and a delay will occur. Such delay often occurs when complex multimedia  
10      data are played on a device with limited processing power, such as a mobile  
11      computing device.

12      At decision block 425, a determination is made whether the actual playback  
13      timing of the video data lags the ideal playback timing. The delay calculated at  
14      block 420 may be used to make this determination. Typically, the actual playback  
15      timing lags the ideal playback timing if the delay exceeded a threshold value. The  
16      threshold value may take ordinary system variations into account. In one  
17      embodiment, the average delays from several frames are used to make the  
18      determination at decision block 425. If the actual playback timing does not lag the  
19      ideal playback timing, process 400 continues at decision block 440. If the actual  
20      playback timing lags the ideal playback timing, the process moves to block 430.

21      At block 430, the video data is synchronized where the actual playback  
22      timing is allowed to catch up with the ideal playback timing. An exemplary  
23      process for synchronizing video data will be discussed in more detail in  
24      conjunction with Fig. 5. Briefly stated, a frame-dropping algorithm is used to  
25

1 determine whether a particular frame should be dropped to recover from the delay.  
2 The frame-dropping algorithm is designed to drop frames in such a way that the  
3 frame rates are smoothly varied.

4 At block 440, a determination is made whether to continue processing  
5 video data. If so, process 400 returns to block 413 where the next frame is  
6 processed. In one embodiment, a time delta to the next time that process 400  
7 needs to be executed is associated with the transition from block 440 to block 413.  
8 This delta is used by the controlling process to guarantee that process 400 receives  
9 priority execution when it has requested service. This has the effect of  
10 guaranteeing a more timely execution of process 400 and further improves the  
11 quality of the preferred video playback rate. When the processing is finished,  
12 process 400 ends.

13 Fig. 5 is an operational flow diagram of an example process 500 for  
14 synchronizing video data. Process 500 may be applied to determine whether a  
15 particular frame should be dropped. Moving from a start block, process 500 goes  
16 to block 510 where the average delay (AvgDelay) associated with a current frame  
17 is determined. AvgDelay is calculated by averaging the delays associated with one  
18 or more previous frames. AvgDelay enables the frame rates to smoothly vary.

19 At block 515, the ideal frame rate (IdealFrameRate) associated with the  
20 current frame is determined. At block 520, a frame skip factor (FrameSkipFactor)  
21 is determined. The FrameSkipFactor may be determined by taking into account  
22 AvgDelay, IdealFrameRate, system tolerance, and other relevant parameters.

1       At decision block 525, a determination is made whether FrameSkipFactor  
2 is greater than IdealFrameRate. If not, process 500 moves to block 545 where the  
3 current frame is shown. Process 500 then ends.

4       Returning to decision block 525, if FrameSkipFactor is greater than  
5 IdealFrameRate, the current frame may have to be dropped. At block 530,  
6 IdealFrameRate is added to an iterating parameter (Iterator). Iterator enables  
7 process 500 to take the disposal of prior frames into account in determining  
8 whether the current frame should be dropped.

9       At decision block 535, a determination is made whether Iterator is greater  
10 than or equal to FrameSkipFactor. If Iterator is greater than or equal to  
11 FrameSkipFactor, the current frame does not have to be dropped. The process  
12 continues at block 540 where FrameSkipFactor is subtracted from Iterator. At  
13 block 545, the current frame is shown and process 500 ends.

14       Returning to decision block 535, if Iterator is not greater than or equal to  
15 FrameSkipFactor, the current frame has to be dropped and the process continues at  
16 decision block 550.

17       At decision block 550, a determination is made whether AvgDelay has  
18 reached a significant percentage of (e.g. greater than half of) the maximum delay  
19 permitted in the system (MaxDelay). MaxDelay is a delay that is large enough to  
20 cause an unacceptably low frame rate. Typically, a 300-millisecond delay is  
21 unacceptable and may be used as the MaxDelay. If AvgDelay is not greater than  
22 half of MaxDelay, process 500 continues at block 555 where the current frame is  
23 dropped and the process ends.

1       Returning to decision block 550, if AvgDelay has exceeded half of  
2       MaxDelay, a more aggressive frame dropping algorithm is employed. At block  
3       560, the input queue is searched for an I-frame that is ready to be rendered. At  
4       decision block 565, a determination is made whether an I-frame has been found.  
5       If so, the I-frame is shown at block 570. If no I-frame is ready to be rendered, the  
6       process continues at block 555 where the current frame is dropped. Process 500  
7       then ends.

8       Fig. 6 illustrates an example of an exemplary computer 600 within which  
9       the systems and methods for managing frame rates during multimedia playback  
10       can be either fully or partially implemented. Exemplary computer 600 is only one  
11       example of a computing system and is not intended to suggest any limitation as to  
12       the scope of the use or functionality of the invention. Neither should the computer  
13       600 be interpreted as having any dependency or requirement relating to any one or  
14       combination of components illustrated in the exemplary computer 600. It is to be  
15       understood that the systems and methods described above may be implemented in  
16       many types of electronic device, including electronic devices that do not have  
17       general computing capabilities.

18       Computer 600 can be implemented with numerous other general purpose or  
19       special purpose computing system environments or configurations. Examples of  
20       well known computing systems, environments, and/or configurations that may be  
21       suitable for use include, but are not limited to, personal computers, server  
22       computers, thin clients, thick clients, hand-held or laptop devices, multiprocessor  
23       systems, microprocessor-based systems, set top boxes, programmable or special  
24       purpose consumer electronics, network PCs, minicomputers, mainframe  
25

1 computers, gaming consoles, distributed computing environments that include any  
2 of the above systems or devices, and the like.

3 The components of computer 600 can include, but are not limited to,  
4 processor 602 (e.g., any of microprocessors, controllers, and the like), memory  
5 604, input devices 606, output devices 608, and network devices 610.

6 Computer 600 typically includes a variety of computer-readable media.  
7 Such media can be any available media that is accessible by computer 600 and  
8 includes both volatile and non-volatile media, removable and non-removable  
9 media. Memory 604 includes computer-readable media in the form of volatile  
10 memory, such as random access memory (RAM), and/or non-volatile memory,  
11 such as read only memory (ROM). A basic input/output system (BIOS),  
12 containing the basic routines that help to transfer information between elements  
13 within computer 600, such as during start-up, is stored in memory 604. Memory  
14 604 typically contains data and/or program modules that are immediately  
15 accessible to and/or presently operated on by processor 602.

16 Memory 604 can also include other removable/non-removable,  
17 volatile/non-volatile computer storage media. By way of example, a hard disk  
18 drive may be included for reading from and writing to a non-removable, non-  
19 volatile magnetic media; a magnetic disk drive may be included for reading from  
20 and writing to a removable, non-volatile magnetic disk (e.g., a “floppy disk”); and  
21 an optical disk drive may be included for reading from and/or writing to a  
22 removable, non-volatile optical disk such as a CD-ROM, DVD, or any other type  
23 of optical media.  
24  
25

1       The disk drives and their associated computer-readable media provide  
2 non-volatile storage of computer-readable instructions, data structures, program  
3 modules, and other data for computer 600. It is to be appreciated that other types  
4 of computer-readable media which can store data that is accessible by computer  
5 600, such as magnetic cassettes or other magnetic storage devices, flash memory  
6 cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random  
7 access memories (RAM), read only memories (ROM), electrically erasable  
8 programmable read-only memory (EEPROM), and the like, can also be utilized to  
9 implement exemplary computer 600.

10       Any number of program modules can be stored in memory 604, including  
11 by way of example, an operating system 626, application programs 628, and data  
12 632. Each of such operating system 626 and application programs 628 may  
13 include an embodiment of the systems and methods for managing frame rates  
14 during multimedia playback. As shown in Fig. 6, application programs 628  
15 include multimedia player 100, which includes Synchronizer 135. Data 632  
16 include multimedia data 105.

17       Computer 600 can include a variety of computer-readable media identified  
18 as communication media. Communication media typically embodies  
19 computer-readable instructions, data structures, program modules, or other data in  
20 a modulated data signal such as a carrier wave or other transport mechanism and  
21 includes any information delivery media. The term “modulated data signal” refers  
22 to a signal that has one or more of its characteristics set or changed in such a  
23 manner as to encode information in the signal. By way of example, and not  
24 limitation, communication media includes wired media such as a wired network or  
25

1 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
2 other wireless media. Combinations of any of the above are also included within  
3 the scope of computer-readable media.

4 A user can enter commands and information into computer 600 via input  
5 devices 606 such as a keyboard and a pointing device (e.g., a “mouse”). Other  
6 input devices 606 may include a microphone, joystick, game pad, controller,  
7 satellite dish, serial port, scanner, touch screen, touch pads, key pads, and/or the  
8 like. Output devices 608 may include a CRT monitor, LCD screen, speakers,  
9 printers, and the like.

10 Computer 600 may include network devices 610 for connecting to  
11 networks, such as local area network (LAN), wide area network (WAN), wireless  
12 phone network, and the like.

13 Although the description above uses language that is specific to structural  
14 features and/or methodological acts, it is to be understood that the invention  
15 defined in the appended claims is not limited to the specific features or acts  
16 described. Rather, the specific features and acts are disclosed as exemplary forms  
17 of implementing the invention.